UNIVERSITY OF MARYLAND/CONQUEST: MUC-4 TEST RESULTS AND ANALYSIS

James Mayfield
Computer Science Department
University of Maryland Baltimore County
Baltimore, MD 21228-5398 USA
mayfield@cs.umbc.edu

SCORES

Only a subset of the MUC-4 slots were generated by the UM/ConQuest ICTOAN system. Our scores for the slots we attempted to fill are shown in Figure 1.

LIMITING FACTORS

The most significant limiting factor we experienced was development time. The ICTOAN system was written from scratch during the first five months of 1992; only the template generation software was reused from our MUC-3 system. The effect of the short development time was compounded by our selection of C as the programming language for the project. We chose C for its speed, and to gain leverage from existing ConQuest software. However, because C is such a low-level language, development of a piece of code in C takes longer than development of equivalent code in a higher-level language such as LISP.

Recall scores for the evaluation system were uniformly poor. The main factors contributing to the low recall scores were:

- The evaluation system did not attempt to fill all template slots. In particular, we did not attempt to guess any slot values that were not known, even when guessing would lead to a significant partial score (*e.g.* in the data slot). The following slots were not filled:

2. INCIDENT: DATE	17. PHYS TGT: TOTAL NUMBER
3. INCIDENT: LOCATION[1]	18. HUM TGT: NAME
11. PERP: ORGANIZATION CONFIDENCE	21. HUM TGT: NUMBER
14. PHYS TGT: NUMBER	22. HUM TGT: FOREIGN NATION
15. PHYS TGT: FOREIGN NATION	23. HUM TGT: EFFECT OF INCIDENT
16. PHYS TGT: EFFECT OF INCIDENT	24. HUM TGT: TOTAL NUMBER

[1] This slot was filled once inadvertentally due to a bug in the semantic net.

SLOT	REC	PRE	OVG	FAL
0. MESSAGE: ID	10	24	76	
4. INCIDENT: TYPE	9	22	76	4
5. INCIDENT: STAGE OF EXECUTION	10	24	76	10
6. INCIDENT: INSTRUMENT ID	0	0	100	
7. INSTRUMENT TYPE	0	0	100	0
8. PERP: INCIDENT CATEGORY	12	35	65	6
9. PERP: INDIVIDUAL ID	3	12	75	
10. PERP: ORGANIZATION ID	9	20	65	
12. PHYS TGT: ID	2	33	67	
13. PHYS TGT: TYPE	0	0	50	0
19. HUM TGT: DESCRIPTION	2	50	0	
20. HUM TGT: TYPE	1	17	17	0
INCIDENT TOTAL	4	19	80	
PERPETRATOR TOTAL	6	22	68	
PHYSICAL TARGET TOTAL	0	11	56	
HUMAN TARGET TOTAL	1	33	8	
MATCHED/MISSING	2	66	5	
MATCHED/SPURIOUS	21	21	70	
MATCHED ONLY	21	66	5	
ALL TEMPLATES	2	21	70	
SET FILLS ONLY	4	75	3	0
STRING FILLS ONLY	3	50	10	
TEXT FILTERING	32	81	19	14

Figure 1: UM/Conquest Scores for TST3

- No grammar rules were in place for performing semantic analysis of nominalized attack actions, or for interpretation of passive verbs. Therefore, the only attacks that were interpreted were ones that were described by active verbs. A low percentage of the reported attacks were described in this way in the test corpora.
- No merging of co-referential phrases or sentences was performed. Thus, two interpretable descriptions of the same attack always led to the generation of two different templates. In addition to increasing overgeneration, this had the effect of reducing recall for slots that were filled in templates that were eventually discarded by the scoring program.
- Although significant hooks are in place in the system for discourse segmentation, the lack of merging of co-referential items made discourse segmentation algorithms irrelevant for the purposes of the evaluation.

ALLOCATION OF RESOURCES

Most of the development time for the ICTOAN system was spent on writing the basic system code and on developing the semantic net of world knowledge. Relatively little time was spent in grammar-writing. Because we had the Proximity Linguistic System available for our dictionary, little time was spent on dictionary development. This turned out to be problematic, because the dictionary contained many obscure or questionable word senses. For example, the word 'the' was listed as an adverb (as in 'the higher the fewer'). Dictionary cleanup is the task we would most like to redo.

SUCCESSES

The two main successes of the ICTOAN system were its speed and its flexibility.

Speed

The system is quite fast. For example, the system processed the 100 stories in the TST3 corpus in under twenty minutes on a lightly-loaded Sun Sparc II with 24MB of memory. The evaluation system had a number of known memory leaks. Because of these leaks, we chose to break up the input texts into individual files and run ICTOAN separately on each. Given time to eliminate these leaks, we believe that the system is capable of processing 100 texts on the same platform in under twelve minutes.

One result made possible by the system's speed is that the ICTOAN system does *no skimming*. That is, the system does not look for 'hot' areas of the text and concentrate its processing power on those areas. Rather, the system processes every sentence of every story to the best of its abilities.

Flexibility

The ICTOAN system architecture is best viewed as a set of parallel streams of data flowing through a pipeline of processes. This stream-based architecture makes it possible to easily integrate top-down and bottom-up processing. Bottom-up processes can be written to read data from low-level streams and build items to be placed on high-level streams. Top-down processes can be written to search for desired elements on low-level streams, based on the items that appear in high-level streams.

Since individual processes may be placed in any order along the pipeline, it is easy to explore the interactions between for example top-down and bottom-up processes. The user needs only to modify a configuration file that indicates which processes should be used and in what order. Furthermore, the overall effectiveness of a particular process can be evaluated by comparing the performance of the system running with the component of interest against the performance of the system running without that component.

We believe that the flexibility of the system will make it easily adaptable to other uses. For example, we intend to use all of the system except the template generator as an automatic text-to-hypertext conversion system [1].

References

[1] James Mayfield and Charles Nicholas. Using semantic nets to enrich hypertext links. Technical Report CS-92-02, University of Maryland Baltimore County, Baltimore, MD, 1992.